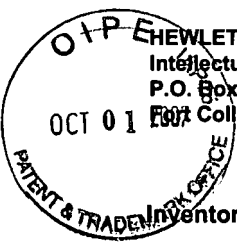


AF/IFW



HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
1801 Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200309109-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Carlos BONILLA

Confirmation No.: 6175

Application No.: 10/628,960

Examiner: T C. Dao

Filing Date: July 28, 2003

Group Art Unit: 2192

Title: AN EMULATION AND NATIVE LANGUAGE INTERFACE TESTING SYSTEM AND METHOD

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on 7/27/2007.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) \$500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

☐ (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

☐ 1st Month
\$120

☐ 2nd Month
\$450

☐ 3rd Month
\$1020

☐ 4th Month
\$1590

☐ The extension fee has already been filed in this application.

☒ (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of \$500. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

☒ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:
Commissioner for Patents, Alexandria, VA 22313-1450

Date of Deposit: 09/27/2007

OR

☐ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571)273-8300.

Date of facsimile:

Typed Name:

Signature:

Ilene L. Fish

Respectfully submitted,

Carlos BONILLA

By

John P. Wagner, Jr.

Attorney/Agent for Applicant(s)

Reg No. : 35,398

Date : 09/27/2007

Telephone : 408-377-0500



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Appellant: Bonilla Patent Application
Application No.: 10/628,960 Group Art Unit: 2192
Filed: July 28, 2003 Examiner: Dao, Thuy Chan
For: AN EMULATION AND NATIVE LANGUAGE INTERFACE TESTING
SYSTEM AND METHOD

APPEAL BRIEF

10/02/2007 EAREGAY1 00000007 000000 10000000
01 FC:1402 00 00 DA



Table of Contents

	<u>Page</u>
Real Party in Interest	1
Related Appeals and Interferences	2
Status of Claims	3
Status of Amendments	4
Summary of Claimed Subject Matter	5
Grounds of Rejection to Be Reviewed on Appeal	7
Argument	8
Conclusion	21
Appendix – Clean Copy of Claims on Appeal	22
Appendix – Evidence Appendix	26
Appendix – Related Proceedings Appendix	27



I. Real Party in Interest

The assignee of the present application is Hewlett-Packard Development Company,
L.P.

II. Related Appeals and Interferences

There are no related appeals or interferences known to the Appellant.

III. Status of Claims

Claims 1-20 are rejected. This Appeal involves Claims 1-20.

IV. Status of Amendments

All proposed amendments have been entered. An amendment subsequent to the Final Action has not been filed.

V. Summary of Claimed Subject Matter

Independent Claim 1 of the present application pertains to an emulation and native language interface test method; independent Claim 5 of the present application pertains to a Java Native Language Interface testing system; and independent Claim 9 of the present application pertains to a Java Native Language Interface Testing method.

In Claim 1, “an emulation and native language interface test method” is recited. This embodiment is depicted in Figure 1A which is described at least on page 7, line 17 - page 9, line 20. For example, “initializing an emulation language virtual machine,” as recited in Claim 1, is described at least in step 11 of method 10 and on page 7, line 6 - page 8, line 3. “[W]rapping native language code in a simulation test macro which creates simulated interfacing problems,” as recited in Claim 1, is described at least in step 12 of method 10 and on page 8, lines 5-13. “[E]xamining reaction to said simulated interfacing problems when an emulation language application is run,” as recited in Claim 1, is described at least in step 13 of method 10 and on page 8, lines 15-20.

In Claim 5, “a Java Native Language Interface testing system” is recited. This embodiment is depicted in Figure 4 which is described at least on page 14, line 27 - page 15, line 17. For example, a “means for communicating information,” as recited in Claim 5, is described at least in Figure 4 (407, 453, 455, and/or 459); on page 15, lines 9-10 (communication bus 407); on page 15, lines 14-15 (input component 453); on page 15, lines 15-16 (display module 455); and on page 15, lines 16-17 (network communication port 459). A “means for processing said information, including instructions for testing a Java Native Language Interface, said means for processing said information coupled to said means for communicating information,” as recited in Claim 5, is described at least in Figure 4 (processor 451) and on page 15, lines 10-12 (processor 451). A “means for storing said information, including said instructions for testing said Java Native Language Interface, said means for storing said information coupled to said means for communicating information,” as

recited in Claim 5, is described at least in Figure 4 (452 and/or 454) and on page 15, lines 12-14 (memory 452 and/or bulk storage 454).

In Claim 9, “a Java Native Language Interface Testing method” is recited. This embodiment is depicted in Figure 1B which is described at least on page 8, line 22 - page 9, line 19; and on page 11, lines 15-21. For example, “investigating Java Native Language Interface test mode status,” as recited in Claim 9, is described at least in step 110 of method 100 and on page 9, lines 4-10. “[R]unning a Java application with simulated Java Native Language Interface problems if said Java Native Language Interface test mode is enabled,” as recited in Claim 9, is described at least in step 120 of method 100 and on page 9, lines 12-19. “[I]nitiating a call to a Java Native Language Interface function directly without said simulated Java Native Language Interface problems if said Java Native Language Interface test mode is not enabled,” as recited in Claim 9, is described at least in step 130 of method 100 and on page 11, lines 15-21.

VI. Grounds of Rejection to Be Reviewed on Appeal

1. Claims 1-4 are rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,826,746 by Evans et al., hereinafter referred to as "Evans."
2. Claims 5-8 are rejected under 35 U.S.C. §102(e) as being anticipated by Evans.
3. Claims 9-20 are rejected under 35 U.S.C. §102(e) as being anticipated by Evans.

VII. Argument

1. Whether Claims 1-4 are Anticipated Under 35 U.S.C. §102(e) by Evans.

Claims 1-4 are rejected under 35 U.S.C. §102(e) as being anticipated by Evans.

Appellant has reviewed the cited art and respectfully submits that the embodiments as recited in Claims 1-4 are not anticipated by Evans in view of the following rationale.

According to the Federal Circuit, “[a]nticipation requires the disclosure in a single prior art reference of each claim under consideration” (W.L. Gore & Assocs. v. Garlock Inc., 721 F.2d 1540, 220 USPQ 303, 313 (Fed. Cir. 1983). “A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference” (Verdegaal Bros. v. Union Oil Co. of California, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987); see also MPEP 2131). However, it is not sufficient that the reference recite all the claimed elements. As stated by the Federal Circuit, the prior art reference must disclose each element of the claimed invention “arranged as in the claim” (emphasis added; Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co., 730 F.2d 1452, 221 USPQ 481, 485 (Fed. Cir. 1984). In other words “[t]he identical invention must be shown in as complete detail as is contained in the ... claim” (emphasis added; Richardson v. Suzuki Motor Co., 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989); see also MPEP 2131). “The elements must be arranged as required by the claim” (MPEP 2131).

A. Cited Art is not “Arranged as in the Claim”

Appellant respectfully submits that Evans does not anticipate the claimed embodiments because the citations relied on by the Rejection of 4/30/2007 (hereinafter “Rejection”) to support anticipation of Claim 1 are not “arranged as in the claim.” In particular, Appellant respectfully asserts that the Rejection has not satisfied a *prima facie* case of anticipation because the Rejection has improperly combined different embodiments of Evans in supporting the anticipation rejection. Appellant respectfully points out that the relied upon art does not teach elements of independent Claim 1 as they are arranged in the claim. Instead, the Rejection picks

and chooses information from a variety of locations and embodiments/examples within the Evans art in an attempt to assemble the elements of Claim 1.

Claim 1 is directed to an emulation and native language interface test method. Yet, in an attempt to establish anticipation of the elements of the method of independent Claim 1, the Rejection parses information from a cherry picked assortment of components, processes, and embodiments. For example, the Rejection parses information regarding a component (Fig 1, JVM 16, and col. 5, lines 7-12) in an attempt to teach a first element of the method of Claim 1. The Rejection then parses information from the background and related art section of Evans (col. 1, lines 1-7 and col. 2, lines 15-22) in an attempt to teach a second element of Claim 1. Finally, the Rejection parses information from the related art section of Evans (col. 2, lines 33-47) and combines this with Figure 6 (event handlers 61-67) in an attempt to teach a third element of the method of Claim 1. Appellant submits that these portions of parsed information are not part of a single method recited by Evans, and do not appear in Evans arranged as required by Claim 1.

Appellant submits that such parsing of Evans is improper and does not satisfy the *prima facie* showing required for establishing anticipation since, by relying on such parsing, the Rejection fails to comply with the requirement of *In re Bond*, cited above, that “[t]he elements must be arranged as required by the claim.”

Therefore, Appellant submits that independent Claim 1 is not anticipated by Evans as the Rejection fails to establish a *prima facie* case for anticipation of Claim 1. As such, Appellant submits that independent Claim 1 is in condition for allowance. Dependant Claims 2-4 depend from Claim 1 which is allowable over Evans. Hence, it is respectfully submitted that dependent Claims 2-4 are patentable over Evans for the reasons discussed above, and are in condition for allowance by virtue of their dependence upon an allowable base claim.

B. Claimed Features are not Met by the Cited Art

Appellant respectfully submits that Evans does not meet or teach the claimed embodiments in the manner set forth in independent Claim 1. Attention is directed to Claim 1 which recites (emphasis added):

An emulation and native language interface test method comprising:
initializing an emulation language virtual machine;
wrapping native language code in a simulation test macro
which creates simulated interfacing problems; and
examining reaction to said simulated interfacing problems
when an emulation language application is run.

Claims 2-4 depend from Claim 1 and recite further features of the claimed embodiment.

Claim 1 recites the feature of “wrapping native language code in a simulation test macro which creates simulated interfacing problems,” emphasis added. The Rejection contends that this feature is taught at col. 1, lines 7-11 and col. 2, lines 15-22 of Evans. However, per Appellant’s understanding, col. 1, lines 7-11 merely recites that the field of the invention of Evans “... generally relates to computer software development tools and a method and system for debugging a Java application that includes native method dynamic load libraries (e.g., C or C++ code).” Per Appellant’s understanding, this is a statement that provides no teaching, and certainly does not teach the feature of “wrapping native language code in a simulation test macro which creates simulated interfacing problems,” which is recited in Claim 1. This cited portion mentions the concept of debugging. However, this cited portion mentions nothing about a “simulation test macro” or about creating “simulated interfacing problems.”

With respect to col. 2, lines 15-22 of Evans, Appellant understands that this cited portion may describe a probe that is “...implemented as two distinct processes, a first ‘daemon’ process that performed native method debugging, and a second ‘probe’ process that performed the Java method debugging... to facilitate the simultaneous debugging of the Java

and C/C++ code comprising the target application.” However, per Appellant’s understanding, simultaneous debugging of Java code and a C/C++ code is not the same as, and does not teach, “wrapping native language code in a simulation test macro which creates simulated interfacing problems,” as is recited in Claim 1. For example, nothing about a “simulation test macro” is mentioned. Moreover, nothing is mentioned about creating “simulated interfacing problems.”

Moreover, Claim 1 recites the feature of “examining reaction to said simulated interfacing problems when an emulation language application is run,” emphasis added. The Rejection contends that this feature is taught in Figure 6 (tool 41 and event handlers 61-67) and at col. 2, lines 33-47 of Evans. However, per Appellant’s understanding, Evans contains no teaching that tool 41 of Figure 6 and/or event handlers 61-67 are capable of examining a reaction to a simulated interfacing problem. While tool 41 and event handlers 61-67 of Evans may be operable to function with a Java Virtual Machine (see, col. 6, line 58 - col. 7, line 47 of Evans), Appellant submits that this is different than, and does not teach, “examining reaction to said simulated interfacing problems when an emulation language application is run,” as is recited in Claim 1.

Additionally, per Appellant’s understanding col. 2, lines 33-47 of Evans may describe a “GUI to permit source-level debugging of JAVA applications... to set breakpoints, execute and step... applications, and examine the application’s stack and variables all whit the click of a mouse.” However, per Appellant’s understanding these are only typical debugging actions and are very different than, and do not teach, “examining reaction to said simulated interfacing problems when an emulation language application is run,” (emphasis added) as is recited in Claim 1.

In view of the claim features not being met by Evans, Appellant respectfully submits that independent Claim 1 overcomes the cited art and is therefore allowable over the 35

U.S.C. §102(e) rejection to Evans. Therefore, Appellant respectfully submits that Evans does not anticipate the additional claimed features as recited in Claims 2-4 that depend from independent Claim 1, and that these claims also overcome the rejection under 35 U.S.C. §102(e) by virtue of their dependence upon an allowable independent claim.

2. Whether Claims 5-8 are Anticipated Under 35 U.S.C. §102(e) by Evans.

Claims 5-8 are rejected under 35 U.S.C. §102(e) as being anticipated by Evans.

Appellant has reviewed the cited art and respectfully submits that the embodiments as recited in Claims 5-8 are not anticipated by Evans in view of the following rationale.

A. Cited Art is not “Arranged as in the Claim”

Appellant respectfully submits that the Rejection has failed to establish a *prima facie* case for anticipation of Claim 5. As indicated above, to anticipate a claim, the reference must teach every element of the claim ...and the elements must be arranged as required by the claim. Appellant respectfully points out that the relied upon art does not teach elements of independent Claim 5 as they are arranged in the claim.

Claim 5 is directed to a Java Native Language Interface testing system. Yet, in an attempt to establish anticipation of the elements of the system of independent Claim 5, the Rejection parses information from a cherry picked assortment of components, processes, and embodiments which are not presented in Evans as arranged in Claim 5. For example, the Rejection parses information regarding an interface code analysis tool 41 and a set of event handlers (See Figure 6 of Evans) and combines this with a description of a Graphic User Interface (col. 2, lines 33-47 of Evans) in an attempt to teach a first element of the system of Claim 5. The Rejection then parses a description regarding the “field of the invention” (col. 1, lines 7-11 of Evans) combined with information about a method/process which is described in the related art section of Evans (col. 2, lines 15-22) in an attempt to teach a second element of the system of Claim 5. Finally, the Rejection parses information regarding a description of “Launching the Application” (see, col. 5, line 61 and col. 6, lines 52-57 of Evans) and combines this with information regarding a description of “obtaining threads” (see, col. 8, line 9, col. 9, line 49; col. 8, lines 15-19; col. 8, line 25 - col. 9, line 67, and Figure 13 of Evans) in an attempt to teach a third element of the method of Claim 5. Appellant submits that these parsed portions

of information are not part of a single system embodiment recited by Evans, and do not appear in Evans as arranged in Claim 5 (e.g., coupled together to form a system).

Appellant submits that such parsing of Evans is improper and does not satisfy the *prima facie* showing required for establishing anticipation since, by relying on such parsing, the Rejection fails to comply with the requirement of In re Bond, cited above, that “[t]he elements must be arranged as required by the claim.”

Therefore, Appellant submits that independent Claim 5 is not anticipated by Evans as the Rejection fails to establish a *prima facie* case for anticipation of Claim 5. As such, Appellant submits that independent Claim 5 is in condition for allowance. Dependant Claims 6-8 depend from Claim 5 which is allowable over Evans. Hence, it is respectfully submitted that dependent Claims 6-8 are patentable over Evans for the reasons discussed above, and are in condition for allowance by virtue of their dependence upon an allowable base claim.

B. Claimed Features are not Met by the Cited Art

Appellant respectfully submits that Evans does not meet or teach the claimed embodiments in the manner set forth in independent Claim 5. Attention is directed to Claim 5 which recites (emphasis added):

A Java Native Language Interface testing system comprising:
means for communicating information;
means for processing said information, including instructions
for testing a Java Native Language Interface, said means for processing
said information coupled to said means for communicating
information; and
means for storing said information, including said instructions
for testing said Java Native Language Interface, said means for storing
said information coupled to said means for communicating
information.

Claims 6-8 depend from Claim 5 and recite further features of the claimed embodiment.

Claim 5 recites the feature of a “means for storing said information, including said instructions for testing said Java Native Language Interface, said means for storing said information coupled to said means for communicating information,” emphasis added. The Rejection contends that this feature is taught at col. 6, lines 52-57; col. 8, lines 15-19; col. 8, line 25 - col. 9, line 67; and Fig 13 of Evans. However, per Appellant’s understanding, col. 6, lines 52-57 of Evans only describes that events are provided to event handlers, that a probe provides code to check constantly or poll for new modules and threads, and that the probe can send a “module loaded” event. This may teach sending and receiving information, but per Appellant’s understanding falls well short of teaching a “means for storing said information,” as recited in Claim 5.

With respect to col. 8, lines 15-19 of Evans, Appellant understands that this cited portion may teach that a debugger reports exceptions. However, Appellant submits that reporting exceptions falls well short of teaching a “means for storing said information,” as recited in Claim 5.

With respect to col. 8, line 25 - col. 9, line 67 of Evans, Appellant understands that this cited portion may teach an “exception filtering process,” “loading of a class,” and “an ability to obtain the status of any application thread.” However, Appellant submits that these concepts fall well short of teaching a “means for storing said information,” as recited in Claim 5.

With respect to Fig. 13 of Evans, Appellant understands that this figure may teach a virtual machine object 45 that includes the ability to send and receive data to an application (including buffered writer 133 and buffered reader 134). Per the description of Figure 13 (see col. 8, line 64 - col. 9, line 15) buffered writer 133 seems capable of performing a process “in order to write data to an application” and buffered reader 134 seems capable of performing another method to “retrieve data from the application.” See col. 9, lines 3-15 of Evans. Per

Appellant's understanding, it appears that a first data is being "written" to an application and a second (different) data is "read" from the application. However, Appellant submits that this does not teach storage of information, let alone a "means for storing said information," as recited in Claim 5. In fact there is no indication that data "written" to an application is being stored; rather, the actions described in col. 8, line 64 - col. 9, line 15 of Evans appear to be more akin to simply sending data to an application.

In view of at least one claim feature not being met by Evans, Appellant respectfully submits that independent Claim 5 overcomes the cited art and is therefore allowable over the 35 U.S.C. §102(e) rejection to Evans. Therefore, Appellant respectfully submits that, for at least the reasons cited above, Evans does not anticipate the additional claimed features as recited in Claims 6-8 that depend from independent Claim 5, and that these claims also overcome the rejection under 35 U.S.C. §102(e) by virtue of their dependence upon an allowable independent claim.

3. Whether Claims 9-20 are Anticipated Under 35 U.S.C. §102(e) by Evans.

Claims 9-20 are rejected under 35 U.S.C. §102(e) as being anticipated by Evans. Appellant has reviewed the cited art and respectfully submits that the embodiments as recited in Claims 9-20 are not anticipated by Evans in view of the following rationale.

A. Cited Art is not “Arranged as in the Claim”

Appellant respectfully submits that the Rejection has failed to establish a *prima facie* case for anticipation of Claim 9. As indicated above, to anticipate a claim, the reference must teach every element of the claim ...and the elements must be arranged as required by the claim. Appellant respectfully points out that the relied upon art does not teach elements of independent Claim 9 as they are arranged in the claim.

Claim 9 is directed to a Java Native Language Interface test method. Yet, in an attempt to establish anticipation of the elements of the method of independent Claim 9, the Rejection parses information from a cherry picked assortment of components, processes, and embodiments which are not presented in Evans as arranged in Claim 9. For example, the Rejection parses a description of a process for launching an application (col. 5, line 59 - col. 6, line 5 of Evans) in an attempt to teach a first element of the method of Claim 9. The Rejection then parses a description regarding the “field of the invention” (col. 1, lines 7-11) combined with information about a method/process which is described in the related art section of Evans (col. 2, lines 15-22), and information regarding running an Interactive Code Analysis Tool Debugger on a single computer system (col. 5, lines 7-12) in an attempt to teach a second element of the method of Claim 9. Finally, the Rejection parses information regarding a description of how “Java developers often ‘extend’ a Java application” (see, col. 1, lines 32-37) combined with information about using a Java Virtual Machine to run an application that is being debugged (see, col. 1, line 66 - col. 2, line 5 of Evans) in an attempt to teach a third element of the method of Claim 9. Appellant submits that these parsed teachings are not part of a single method embodiment recited by Evans, and do not appear in Evans as arranged in Claim 9.

Appellant submits that such parsing of Evans is improper and does not satisfy the *prima facie* showing required for establishing anticipation since, by relying on such parsing, the Rejection fails to comply with the requirement of In re Bond, cited above, that “[t]he elements must be arranged as required by the claim.”

Therefore, Appellant submits that independent Claim 9 is not anticipated by Evans as the Rejection fails to establish a *prima facie* case for anticipation of Claim 9. As such, Appellant submits that independent Claim 9 is in condition for allowance. Dependant Claims 10-20 depend from Claim 9 which is allowable over Evans. Hence, it is respectfully submitted that dependent Claims 10-20 are patentable over Evans for the reasons discussed above, and are in condition for allowance by virtue of their dependence upon an allowable base claim.

B. Claimed Features are not Met by the Cited Art

Appellant respectfully submits that Evans does not meet or teach the claimed embodiments in the manner set forth in independent Claim 9. Attention is directed to Claim 9 which recites (emphasis added):

A Java Native Language Interface test method comprising:
investigating Java Native Language Interface test mode status;
running a Java application with simulated Java Native
Language Interface problems if said Java Native Language Interface
test mode is enabled; and
initiating a call to a Java Native Language Interface function
directly without said simulated Java Native Language Interface
problems if said Java Native Language Interface test mode is not
enabled.

Claims 10-20 depend from Claim 9 and recite further features of the claimed embodiment.

Claim 9 recites the feature of “running a Java application with simulated Java Native
Language Interface problems if said Java Native Language Interface test mode is enabled,”
emphasis added. The Rejection contends that this feature is taught at col. 1, lines 7-11; col. 2,

lines 15-22; and col. 5, lines 7-12 of Evans. However, per Appellant's understanding, col. 1, lines 7-11 of Evans merely recites that the field of the invention of Evans "... generally relates to computer software development tools and a method and system for debugging a Java application that includes native method dynamic load libraries (e.g., C or C++ code)." Per Appellant's understanding, this is a statement that provides no teaching, and certainly does not teach the feature of "running a Java application with simulated Java Native Language Interface problems if said Java Native Language Interface test mode is enabled," which is recited in Claim 9. This cited portion mentions the concept of debugging. However, this cited portion mentions nothing about a "simulated Java Native Language Interface problems" or about a "Java Native Language Interface test mode."

With respect to col. 2, lines 15-22 of Evans, Appellant understands that this cited portion may describe a probe that is "...implemented as two distinct processes, a first 'daemon' process that performed native method debugging, and a second 'probe' process that performed the Java method debugging... to facilitate the simultaneous debugging of the Java and C/C++ code comprising the target application." However, per Appellant's understanding, simultaneous debugging of Java code and a C/C++ code is not the same as, and does not teach, "running a Java application with simulated Java Native Language Interface problems if said Java Native Language Interface test mode is enabled," as is recited in Claim 9. For example, this cited portion mentions nothing about a "simulated Java Native Language Interface problems." or about a "Java Native Language Interface test mode."

With respect to col. 5, lines 7-12 of Evans, Appellant understands that this cited portion may teach that a "Java Virtual Machine (16) which is running the ICAT2 probe (41)." However, Appellant submits that this is not the same as "running a Java application with simulated Java Native Language Interface problems if said Java Native Language Interface test mode is enabled," as recited in Claim 9. Per Appellant's understanding nothing in this

cited portion or in the Evans art describes or teaches “simulated Java Native Language Interface problems” or a “Java Native Language Interface test mode.”

In view of at least one claim feature not being met by Evans, Appellant respectfully submits that independent Claim 9 overcomes the cited art and is therefore allowable over the 35 U.S.C. §102(e) rejection to Evans. Therefore, Appellant respectfully submits that, for at least the reasons cited above, Evans does not anticipate the additional claimed features as recited in Claims 10-20 that depend from independent Claim 9, and that these claims also overcome the rejection under 35 U.S.C. §102(e) by virtue of their dependence upon an allowable independent claim.

In summary, Appellant respectfully submits that the rejection of the Claims 1-20 does not satisfy the requirements of a *prima facie* case of anticipation, and is thus improper. Additionally, Appellant submits that the cited art does not teach one or more of the claimed features of each of independent Claims 1, 5, and 9, and thus Claims 1-20 are not anticipated by the prior art. Accordingly, Appellant respectfully submits that the rejection of Claims 1-20 under 35 U.S.C. §102(e) should be reversed.

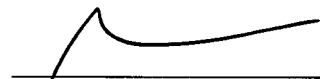
Conclusion

Appellant believes that pending Claims 1-20 are patentable over Evans. As such, Appellant submits that Claims 1-20 are patentable over the prior art.

Appellant respectfully requests that the rejection of Claims 1-20 be reversed. The Appellant wishes to encourage the Examiner or a member of the Board of Patent Appeals to telephone the Appellant's undersigned representative if it is felt that a telephone conference could expedite prosecution.

Respectfully submitted,
Wagner Blecher LLP

Dated: 09/27/2007



John P. Wagner, Jr.
Registration No.: 35,398

Wagner Blecher LLP
Westridge Business Park
123 Westridge Drive
Watsonville, CA 95076
San Jose, CA 95113

Phone: (408) 377-0500
Facsimile: (408) 722-2350

VIII. Appendix - Clean Copy of Claims on Appeal

1. (Original) An emulation and native language interface test method comprising:
initializing an emulation language virtual machine;
wrapping native language code in a simulation test macro which creates simulated interfacing problems; and
examining reaction to said simulated interfacing problems when an emulation language application is run.
2. (Original) An emulation and native language interface test method of Claim 1 wherein said emulation language virtual machine creates a runtime environment and said runtime environment can include a class loader subsystem and an execution engine.
3. (Original) An emulation and native language interface test method of Claim 1 wherein said simulated test problems include simulations of error conditions associated with a native language code method attempt to respond to a call from emulation language code.
4. (Original) An emulation and native language interface test method of Claim 1 further comprising forwarding an indication that there is an insufficient memory allocation exception to a native language method attempting to ascertain an indication of a memory location for information associated with a native language function.
5. (Previously Presented) A Java Native Language Interface testing system comprising:
means for communicating information;
means for processing said information, including instructions for testing a Java Native Language Interface, said means for processing said information coupled to said means for communicating information; and

means for storing said information, including said instructions for testing said Java Native Language Interface, said means for storing said information coupled to said means for communicating information.

6. (Previously Presented) The Java Native Language Interface testing system of claim 5 wherein said means for processing performs a Java Native Language Interface test method.

7. (Previously Presented) The Java Native Language Interface testing system of claim 5 wherein said instructions include an interface testing macro module.

8. (Previously Presented) The Java Native Language Interface testing system of claim 5 emulates a Java Virtual Machine.

9. (Original) A Java Native Language Interface test method comprising:
investigating Java Native Language Interface test mode status;
running a Java application with simulated Java Native Language Interface problems if said Java Native Language Interface test mode is enabled; and
initiating a call to a Java Native Language Interface function directly without said simulated Java Native Language Interface problems if said Java Native Language Interface test mode is not enabled.

10. (Previously Presented) The Java Native Language Interface test method of Claim 9 wherein said Java Native Language Interface test mode status indicator indicates if said Java Native Language Interface test mode status is enabled.

11. (Previously Presented) The Java Native Language Interface test method of Claim 9 wherein said Java Native Language Interface test mode status indicator is a flag wherein a state of said flag indicates if said Java Native Language Interface test mode status is set.

12. (Previously Presented) The Java Native Language Interface test method of Claim 9 wherein a register value indicates said Java Native Language Interface test mode status.
13. (Previously Presented) The Java Native Language Interface test method of Claim 9 wherein said problems include identifying indications of Java Native Language Interface code trouble associated with out of memory situations.
14. (Previously Presented) The Java Native Language Interface test method of Claim 9 wherein a Java Native Language Interface problem simulation process is performed to simulate Java Native Language Interface problems.
15. (Previously Presented) The Java Native Language Interface test method of Claim 9 further comprising:
- determining a Java Native Language Interface problem simulation occurrence level;
 - introducing simulation randomness;
 - performing an analysis whether to initiate a simulation of Java Native Language Interface problem;
 - calling a Java Native Language Interface memory allocation function normally;
 - forwarding a Java Native Language Interface problem indicator automatically; and
 - implementing a reaction to the Java Native Language Interface problem indication.
16. (Previously Presented) The Java Native Language Interface test method of Claim 15 further comprising:
- looking up a predefined Java Native Language Interface problem simulation occurrence level;
 - generating a random value; and

correlating said random value to said Java Native Language Interface problem simulation occurrence level.

17. (Previously Presented) The Java Native Language Interface test method of Claim 16 further comprising:

comparing said randomly generated value to said Java Native Language Interface problem simulation occurrence level; and

initiating a simulation of a Java Native Language Interface problem if said generated value from is less than said Java Native Language Interface problem simulation occurrence level.

18. (Previously Presented) The Java Native Language Interface test method of Claim 16 further comprising initiating a controlled shut down.

19. (Previously Presented) The Java Native Language Interface test method of Claim 16 further comprising clearing a system and canceling information inventory collections that is occupying memory space.

20. (Previously Presented) The Java Native Language Interface test method of Claim 16 further comprising providing an indication of the Java Native Language Interface problem to a user.

IX. Evidence Appendix

No evidence is herein appended.

X. Related Proceedings Appendix

No related proceedings.